Journal of Computer and Communications, 2016, 4, 30-43
Published Online August 2016 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2016.410004

Scientific Research Publishing

Hardware Design of Moving Object Detection on Reconfigurable System

Hung-Yu Chen[1], Yuan-Kai Wang[2]

[1]Graduate Institute of Applied Science and Engineering, Fu Jen Catholic University, Taiwan
[2]Department of Electrical Engineering, Fu Jen Catholic University, Taiwan
Email: ronwisely@islab.tw, ykwang@fju.edu.tw

Received 20 June 2016; accepted 7 August 2016; published 10 August 2016

Abstract

Moving object detection including background subtraction and morphological processing is a critical research topic for video surveillance because of its high computational loading and power consumption. This paper proposes a hardware design to accelerate the computation of background subtraction with low power consumption. A real-time background subtraction method is designed with a frame-buffer scheme and function partition to improve throughput, and implemented using Verilog HDL on FPGA. The design parallelizes the computations of background update and subtraction with a seven-stage pipeline. A stripe-based morphological processing and accounting for the completion of detected objects is devised. Simulation results for videos of VGA resolutions on a low-end FPGA device show 368 fps throughput for only the real-time background subtraction module, and 51 fps for the whole system, including off-chip memory access. Real-time efficiency with low power consumption and low resource utilization is thus demonstrated.

Keywords

Background Substraction, Moving Object Detection, Field Programmable Gate Array (FPGA), Hardware Acceleration

1. Introduction

Moving object detection is typically the first processing stage in video surveillance systems. It is one of the most important steps in smart video surveillance, which aims to detect foreground objects and events. Foreground is essential for tracking objects and maintaining their identities. The detection of foreground objects can be achieved by the representation of scene background. The foreground object is determined by locating significant differences between the current frame and background representation.

How to cite this paper: Chen, H.-Y. and Wang, Y.-K. (2016) Hardware Design of Moving Object Detection on Reconfigurable System. *Journal of Computer and Communications*, **4**, 30-43. http://dx.doi.org/10.4236/jcc.2016.410004

There are many proposed methods for detecting moving objects, such as temporal difference, optical flow and background subtraction [1]. Non-recursive methods including temporal difference and optical flow adopt sliding windows to build background models. This method consists of saving all image frames of each picture into a moving window, and applying some statistic measures such as median filter [2]-[4] or mean filter [5] to analyze the change of each pixel by time in the window screen in order to estimate the background image. The goal of moving windows is to ensure that the background model is at its most up to date condition in order to cut out old pixels and allow the entry of new pixels. However, it does require longer moving windows when dealing with items that move slowly, which in turn requires very large pixel memory space. Background subtraction is more important in present study and has the best accuracy among these methods. Common background subtraction methods include Running Average (RA) [6], Gaussian mixture model (GMM) [7] and nonparametric kernel methods [8]. Although GMM and nonparametric methods are stable algorithms, their complexity [9] [10] makes it impossible to be implemented by hardware approach. RA is also a stable algorithm, but its signal processing is more tractable for hardware implementation because of single modality in statistical modeling.

Figure 1 illustrates a general algorithmic step of background subtraction. $P_{t+1}(x,y)$ is the current real-time image pixel of the location at time $t+1$. $B_{t+1}(x,y)$ is the current pixel location of the established background image. $B_t(x,y)$ is the pixel location of the established background image at time t. $T_{t+1}(x,y)$ is the adaptable threshold value of the location of the object image with noise when time is $t+1$. $M_{t+1}(x,y)$ is the adaptable threshold value, which is input to the Morphology process unit. $\sigma_t+1(x,y)$ is the adaptable threshold value of the location of the background image with morphology process when time is $t+1$.

DSP (Digital Signal Processing), GPU (Graphics Processing Unit) and FPGA (Field Programmable Gate Array) are three hardware solutions used to accelerate background subtraction algorithms. DSP is a powerful and very fast microprocessor, able to achieve real-time digital signal processing. DSP can use instruction-level parallelization technique with multiple paths to achieve speedup, but its computational power is limited [11]. GPUs are very powerful processors which outperform central processing units in special applications on computers and portable devices. GPUs are also capable of parallel computing by placing many threads on massive cores and computing them at the same time. However, they are not power efficient. FPGA is an integrated circuit designed to be configured by a designer and generally specified using a hardware description language (HDL). It is similar to that used for an ASIC (Application-specific Integrated Circuit). FPGA has the ability to handle complex operations in parallel by its reconfigurable capability. This capability combined with pipeline design can speed

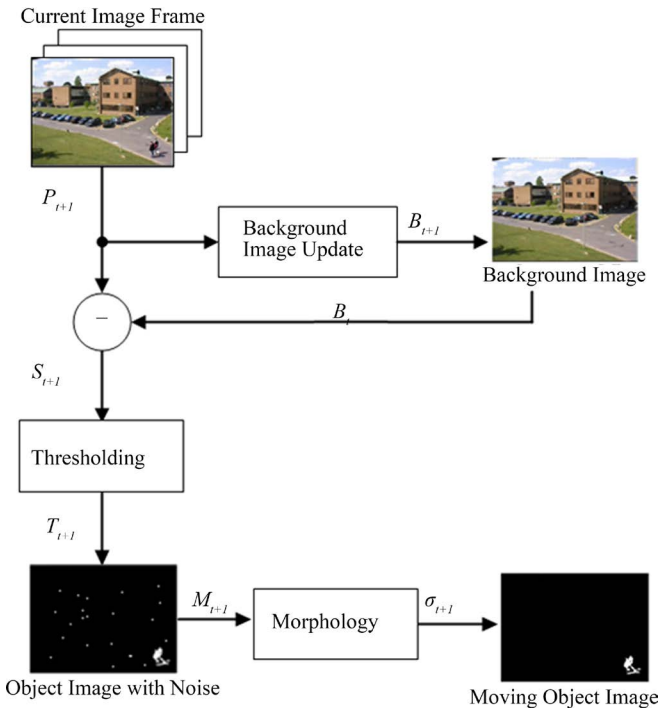

Figure 1. Computational flow of background subtraction.

up background subtraction. The architecture of the FPGA is highly parallel and tailored to efficiently construct image and complex algorithms. FPGAs are therefore suitable for implementations of image processing and computer vision algorithms in embedded systems.

Many studies have been devoted to accelerating moving object detection by FPGA. The following papers are consistently devoted to achieving real-time performance for 640 × 480 or 720 × 576 resolutions with high-end and high-power FPGA. Appiah *et al.* [12] achieved 60 fps by FPGA, which is promising compared with the 25 fps achieved by a 3 GHz Pentium 4. However, the method requires larger hardware resources because four frame buffers must be allocated to handle the process. Cucchiara *et al.* [13] applied four FPGAs with four frame buffers to achieve object detection. Its best performance can only achieve 5 fps, which is far from meeting real-time requirements. Elgammal *et al.* [8] used Wronskian statistics for moving object detection and implemented it on FPGA. However, its maximum performance of 15 fps is not sufficient for real-time detection [14]-[16]. Genovese *et al.* [17] applied an OpenCV GMM algorithm implementation able to process 22 fps on 1920 × 1080 resolution when implemented on Virtex5 FPGA. Jang *et al.* [18] proposed a circuit able to process 1024 × 1024 video sequences at 38 fps when implemented on a VirtexII FPGA platform. Genovese *et al.* [19] applied two hardware implementations of the OpenCV version of the Gaussian Mixture Model (GMM), a background identification algorithm. When implemented on Virtex6, the proposed circuit processed 60 fps on a 1920 × 1080 resolution.

The architecture proposed in this study was designed with a hardware design for running average and morphology algorithms by FPGA. This system is proposed in order to achieve real-time performance with low power consumption. The proposed circuit has been experimentally validated through experimental measurements on a hardware platform.

The main contributions of this paper are the following:

1) An innovative, hardware-oriented formulation of the RA equations that allows hardware speed improvement and saving.

2) A background subtraction and morphology algorithm is proposed and accelerated by reconfigurable hardware which allows embedded systems to operate in real-time.

3) The experimental demonstration of the proposed FPGA circuit in running on-line video systems.

The main block includes five modules as shown in **Figure 2**. The block must convert color format from RGB to YCbCr, establish the background model by Y components, apply the background subtraction to compare the foreground and background's Y components, and finally perform morphology to obtain the result of object detection.

The remainder of this paper is organized as follows. Section 2 reviews the running average and morphology algorithm, and also presents the proposed Real-Time Background Subtraction (RTBS) design. Section 3 presents the proposed RTBS system. Simulation and verification of the design are given in Section 4. Section 5 concludes advantages of the design.

2. Methodology

This section mainly describes the system build algorithms and methods. It first reviews the formulation of the background subtraction and the RTBS algorithm. The RTBS improvement of the RA algorithm by removing division operation is then described. Dataflow of the RTBS is also analyzed. Next, the morphology algorithm and pipeline process are discussed. Dataflow of the morphology is also analyzed. Finally, the pipeline process diagram of morphology is explained.

2.1. Background Subtraction

The background subtraction algorithm consists of two steps: differencing and background modeling. The differencing

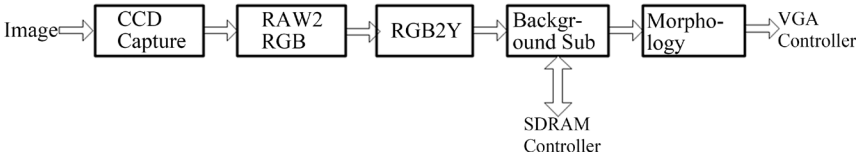

Figure 2. Detailed processing modules of the main block.

step extracts motion pixels by computing the difference between the current frame and the background model. The background model is statistically built with single modality assumption. The differencing step can be formulated as follows:

$$M_{t+1}(x,y) = \left| P_{t+1}(x,y) - B_t(x,y) \right| \tag{1}$$

where $P_{t+1}(x,y)$ is the pixel value of the current frame at time $t+1$, $B_t(x,y)$ is the background model at time t, and $M_{t+1}(x,y)$ is the subtraction result.

The background model of each pixel is assumed to be single Gaussian, and its parameters can be recursively updated by a new frame, which can improve computational efficiency and reduce memory resource allocation [8]. The recursive form of the expected value of the single Gaussian, $B_t(x,y)$, is described as follows:

$$B_t(x,y) = \frac{k}{k+1} B_{t-1}(x,y) + \frac{1}{k+1} P_t(x,y) \tag{2}$$

where $B_t(x,y)$ is the established background image at time t, and k is the learning parameter controlling the background learning speed. The difference image $M_{t+1}(x,y)$ is then thresholded into $T_{t+1}(x,y)$ by an adaptive threshold obtained by recursively updating the variance of the Gaussian model.

$$T_{t+1}(x,y) = \begin{cases} 1, & \text{if } M_{t+1}(x,y) < \lambda \sigma_t(x,y), \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

$$\sigma_t^2(x,y) = \frac{k}{k+1} \sigma_{t-1}^2(x,y) + \frac{1}{k+1} M_t^2(x,y), \tag{4}$$

where standard deviation $\sigma_t(x,y)$ is applied to the adaptive threshold and recursively updated by the current frame. The parameter λ determines the desired precision of thresholding.

The above formulation for background subtraction and updating includes five multiplications, two divisions, and one radical expression operation. The divisions in Equation (2) and Equation (4) require significant logic circuit resources and can slow down computational performance. By applying integer arithmetic to replace the division operation with bit shifting, the hardware design of the RA algorithm is significantly improved. In addition, the variance σ_t^2 obtained in Equation (4) must be square-rooted into standard deviation σ_t for the thresholding in Equation (3). The implicit radical expression must be eliminated. As a result, reformulation is necessary in order to shorten computational time as well as reduce resource consumption.

In order to apply shift circuit instead of division, the denominators of Equation (2) and Equation (4) must be modified. The σ_t^2 is replaced with V_t and the σ_t in Equation (3) is substituted with V_t, where both sides of the condition must be squared. The three new equations are given below:

$$B_t(x,y) = \frac{k}{N} B_{t-1}(x,y) + \frac{(N-k)}{N} P_t(x,y), 0 < k < N, \tag{5}$$

$$V_t(x,y) = \frac{k}{N} V_{t-1}(x,y) + \frac{(N-k)}{N} M_t^2(x,y), 0 < k < N, \tag{6}$$

$$T_{t+1}(x,y) = \begin{cases} 1, & \text{if } M_{t+1}^2(x,y) < \lambda^2 V_t(x,y), \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where N is the m power of 2, *i.e.*, $N = 2^m$. The $k/(k+1)$ of RA is replaced with k/N, and $1/(k+1)$ is replaced with $(N-k)/N$. As a result, shift circuit is applied to replacing division operations. Mathematically, the reformulation produces residue between RA and RTBS on background updating. However, it can be demonstrated that, practically, the residue vanishes as t increases.

Before using Verilog for hardware design, Equation (5) is analyzed in more detail to identify the data flow of background updating. The data flow is shown in **Figure 3(a)**. The data flow of Equation (6) to find the adaptive threshold is shown in **Figure 3(b)**. Equation (7) performs a new adaptive thresholding mechanism to find objects. Its dataflow diagram is shown in **Figure 3(c)**.

Now the required arithmetic operations between RTBS and RA are compared. A detailed comparison is given in **Table 1**. Although RTBS requires two extra multiplications, it eliminates the need for division and radical

expression, which can dramatically reduce hardware resource utilization. **Figure 4** illustrates the residues for $N = 128$. Higher residues may exist for small t, but residues diminish to zero when $t = N$.

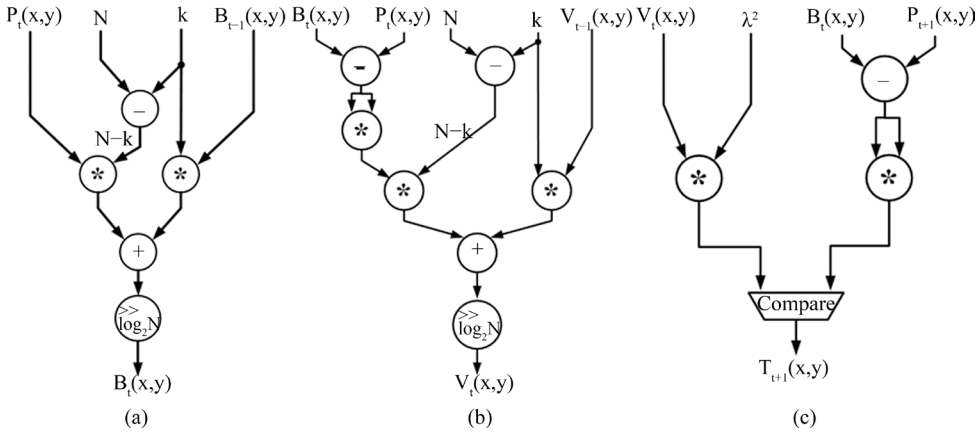

Figure 3. Data flow diagram: (a) Background modeling in Equation (5). (b) Adaptive threshold determination in Equation (6). (c) Foreground extraction in Equation (7).

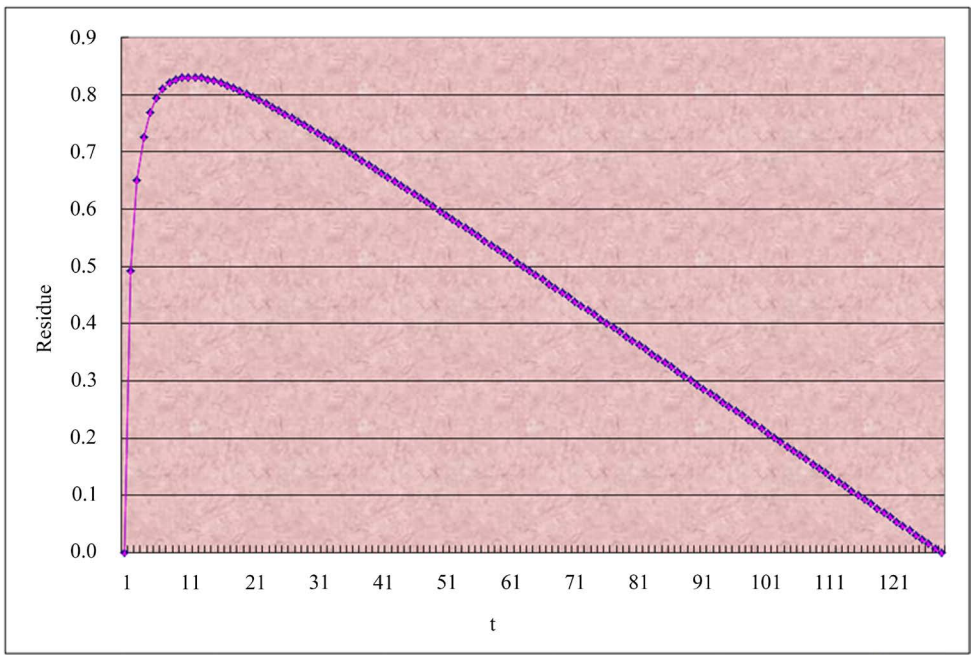

Figure 4. Residue analysis between Equation (2) and Equation (5).

Table 1. Arithmetic operations of running average and the RTBS.

Operation	RA	RTBS
Addition (+)	4	2
Subtraction (−)	2	3
Multiplication (×)	5	7
Division (/)	2	0
Radical Expression (√)	1	0

2.2. Morphology

Morphology theory's hardware designs mainly apply the solutions of corrosion and expansion. Before discussing these two solutions, it is necessary to understand an important step, which is shown in **Figure 5**, called the structuring element. This study mainly uses a 3 × 3 mask, using liner buffer alone with register to get the 9 closest pixel (M1 ~ M9), in which the M5 will be the Origin.

Erosion and dilation are the two basic elements of image handling, and both Opening and Closing will apply these two principles. However, both erosion and dilation must establish the 3 × 3 image window in order to obtain the pixels from M1 ~ M9 and to define P1 ~ P9 as the value in the structuring element. The final number which applies the AND gate is erosion, while the one using the OR gate is expansion (see **Figure 6**).

The following section will describe the flow of the morphology data process. As with the DFG shown in **Figure 7**, a few nodes are first defined as follows:

- "E" is the image window for 3 × 3 for image erosion.
- "D" is the image window for 3 × 3 for dilation.

First, it is necessary to establish a 3 × 3 image window and enter node "E" for erosion followed by node "D" for dilation. After the image is opened for clear up, a 3 × 3 image window is established, and node "D" is entered to handle the erosion, then again, another 3 × 3 image window is established in order to enter "E" for

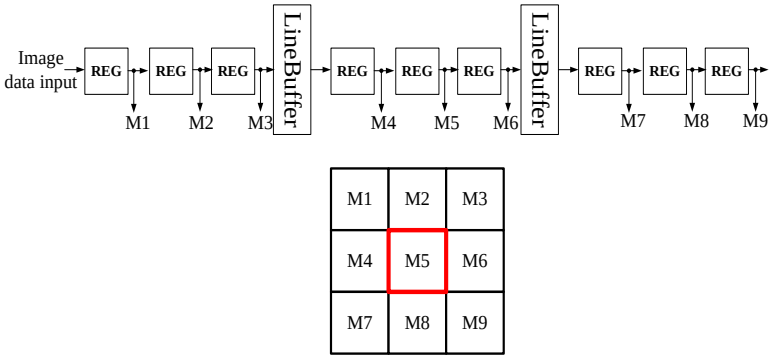

Figure 5. 3 × 3 structuring element.

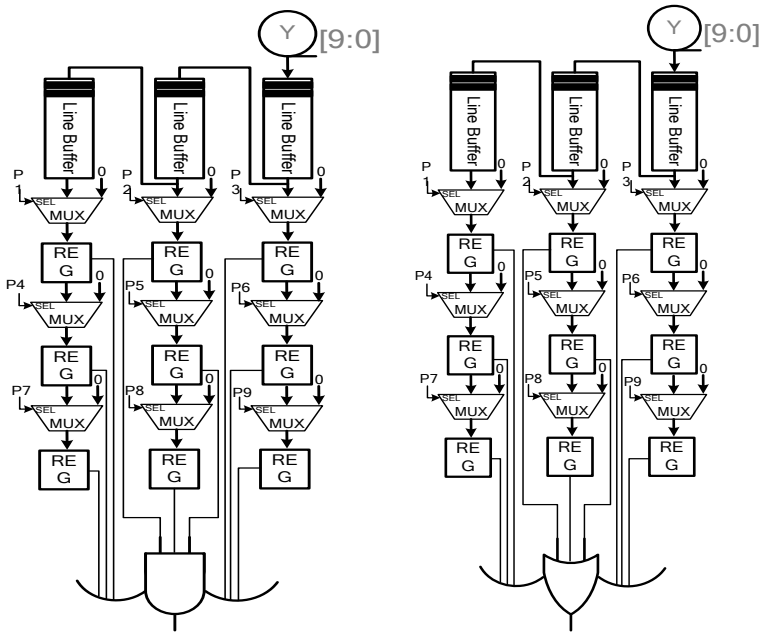

Figure 6. Hardware implementation structure: (a) Erosion, (b) Dilation.

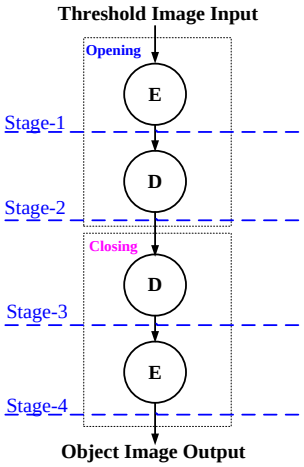

Figure 7. Morphology process DFG.

dilation in order to complete Closing to fix up the image.

In Morphology, a 4-stage pipeline is applied to handle the issue as shown in **Figure 8**.

3. System

This section first reviews the system of the RTBS algorithm. This study applies a Field Programmable Gate Array. **Figure 9** shows the system structure of a moving object detection system.

A 1.3 megapixel CMOS digital module was applied to obtain the image resource, further image handling was conducted by FPGA, the image color scheme was changed from RGB to YCbCr, the background was established by its Y, the Background Subtraction was applied to compare the foreground and background's Y, and moving object detection was finally achieved.

The Real Time Background Subtraction (RTBS) integrated into the system implements Background Subtraction, background updating and adaptive threshold. Thus, the overall operation can be implemented in hardware using subs, adds, shifters and multipliers. The RTBS employs hardware features such as parallelism and pipelining. The architecture is pipelined into 7 stages. In the input image data fetches 2 pixels from the input image port, and forwards them to the input ports of the line buffers (10 bits to each buffer), which have a FIFO structure.

Background Subtraction and Morphology were integrated mainly due to the simple hardware structure, which can be easily parallelized to provide more than one pixel per single cycle. This is important, as the ability of the RTBS to perform parallel computations depends on the ability of the line buffer to provide multiple pixels per cycle. The Morphology employs hardware features such as parallelism and pipelining, in an effort to parallelize the repetitive calculations involved in the Erosion and Dilation operations, and uses optimized memory structures in order to reduce the memory reading redundancy. The architecture of the RTBS is shown in **Figure 10**.

The architecture implementation of the RTBS algorithm consists of a memory controller, RGB to YCbCr color space converters (RGB 2YCvCr) and Background Image storage. The memory controller fetches the RGB color values corresponding to the support first Line Buffer (FIFO Buffer 1) and second Line Buffer (FIFO Buffer 2) in a column-wise fashion (1 pixel value per input image every clock cycle) from the external memory. The architecture of the 4 port SDRAM control block is shown in **Figure 11**.

Those values are then converted to grayscale by the RGB2YCbCr unit, and to their corresponding 10-bit YCbCr representation by the RGB2YCbCr units. The Background image values of RTBS computed by the Background Subtraction unit and Morphology unit are temporarily stored in on-chip SDRAM frame buffers.

The design was simulated and implemented on a low-end FPGA with low power consumption. The FPGA has about five hundred thousand gate counts, 150 18 × 18 multipliers and 60 KB internal memory. The main system clock works at 25 MHz, which has very low power consumption compared with that of a PC with a 2.4 GHz processor. Verilog HDL is adopted for implementation and QUARTUS II for synthesis. An external 64 MB SDRAM memory is used for frame buffers to store background models.

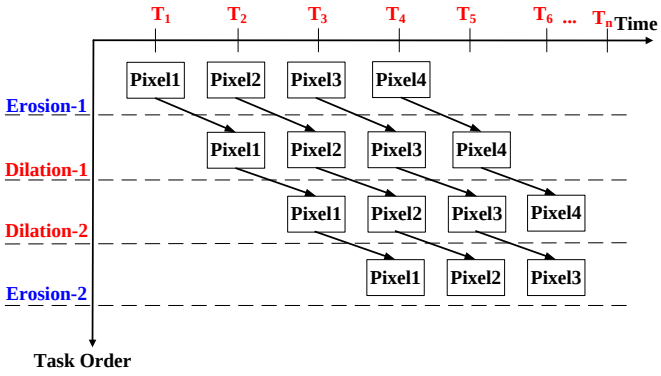

Figure 8. Pipeline process diagram for Morphology.

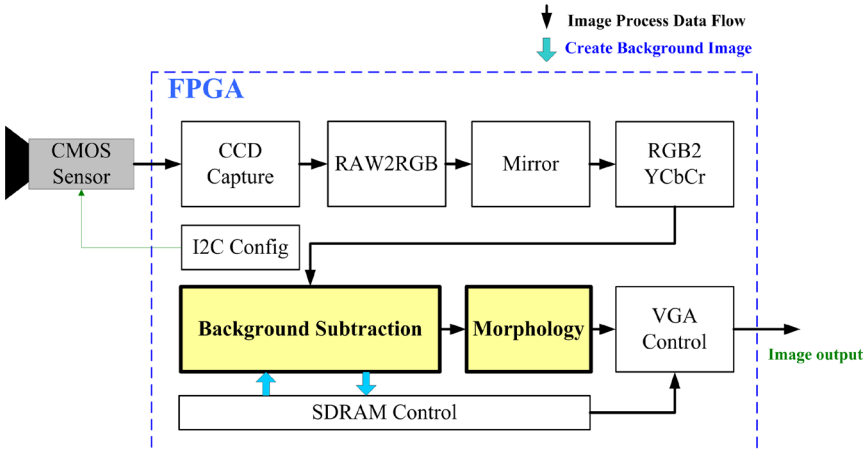

Figure 9. Structure of FPGA moving object detection system.

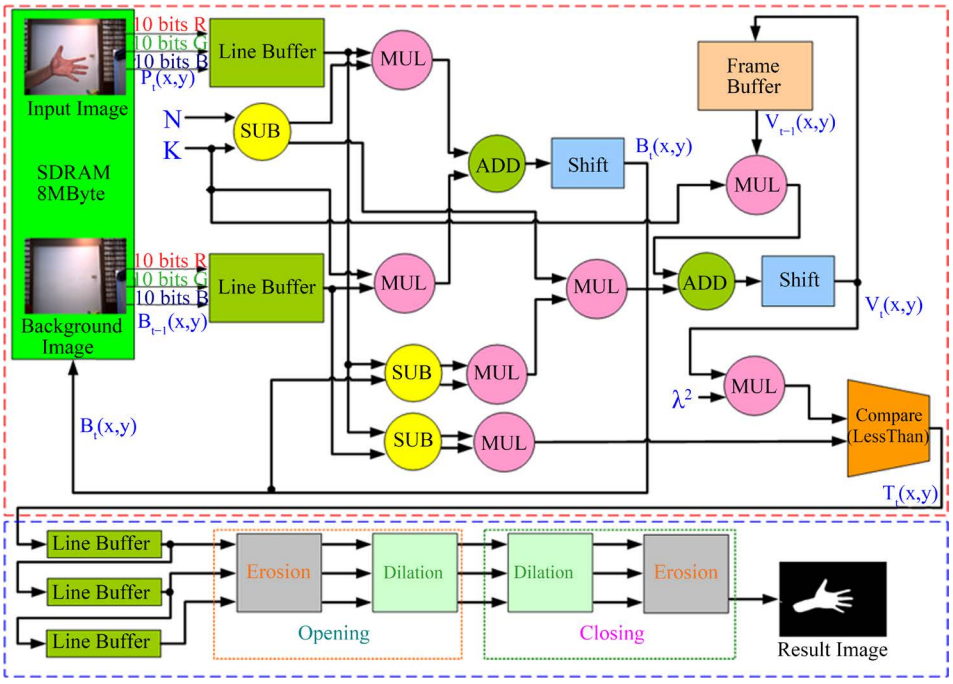

Figure 10. The architecture of the RTBS.

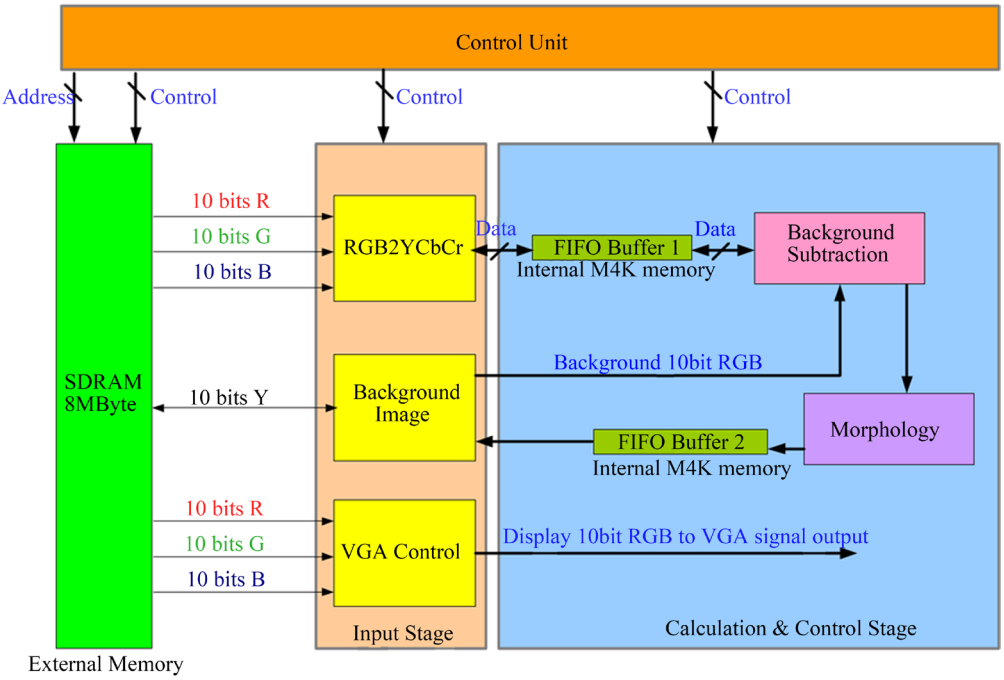

Figure 11. The 4 port SDRAM control block.

A 1.3 megapixel CMOS sensor module is responsible for acquiring raw color images for the FPGA platform. The raw data format comes with a Bayes pattern arrangement, and has to be converted to RGB color format. The system is divided into three blocks, as shown in **Figure 12**. The main block receives raw Bayes data and performs the RTBS background subtraction task. Background models B_t and V_t are stored in the off-chip SDRAM memory because of the limited internal memory of the FPGA.

The subtraction result is sent to the VGA controller for display due to the constraints of peripheral circuits, and the system has two clock rates: 25 MHz for CMOS image acquisition and processing, and 120 MHz for SDRAM access of background models.

4. Simulation and Experiment

First, this section presents the FPGA hardware circuit by experiment image testing to verify its accuracy. Next, Frame Rate is applied in order to analyze and discuss the Throughput. The experiment also applies the results of the BS and RTBS equations to determine the difference between the two after image analysis. The process of the FPGA hardware resource usage is explained later in the article.

4.1. Frame Rate Analysis and Experiment

This section will further verify the performance of the system. A seven-segment display was used to count the processed number of frames within a minute-with an average result of 51 fps. This result differs from the 127 fps result from synthesis and simulation as synthesis and simulation have different clock rates.

Next, the main clock is analyzed. The clock rate in the main block is 25 MHz. Furthermore, each frame will have the waiting blanking time, with the current time of exposure setting. The actual image information clock is around 70%, which equals 25 MHz × 0.70 = 17.5 MHz. It is further calculated that its frame rate is 56 fps (17.5 MHz/0.3072 M).

From the above analysis the result of 56 fps is derived, which is very close to 51 fps. It is also known that the limit of this experiment is the main clock, which is due to the CMOS sensor clock.

Moreover, the RTBS was implemented using a 1.8 GHz P4 CPU with 1GB DDR/333 MHz memory. This yielded a frame rate of 3.22 fps, which differs significantly from the FPGA result. **Table 2** shows more detailed frame rate results, and demonstrates that hardware is far better than software in terms of efficiency.

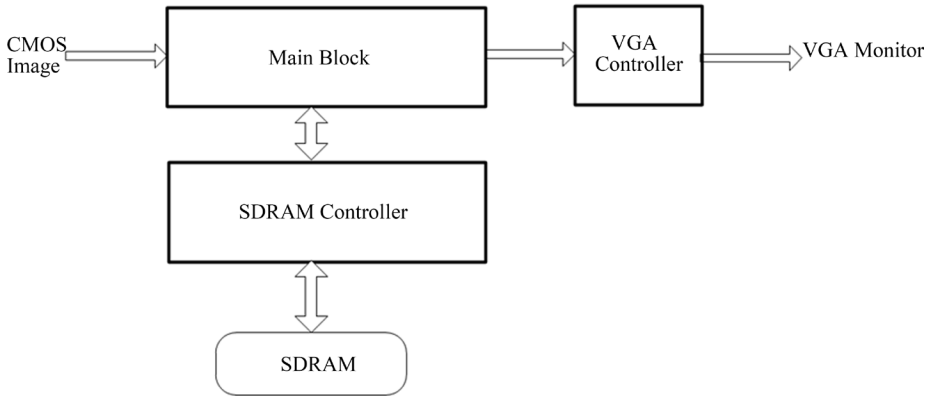

Figure 12. Diagram of system block.

Table 2. Frame rate analysis.

Block	FPGA Simulation	FPGA Verification	PC
Main Block	368	56	–
System Block	127	51	3.22

This study uses QUARTUS II to further analyze the frame rate performance by simulation. The QUARTUS II contains TimeQuest Timing Analyzer that applies industry-standard Synopsys Design Constraint methodology for constraint designs. The timing characteristics and timing performance of our system are obtained from the analyzer. The timing analysis reports that the clock period of the main block is 8.8 ns, which equals 113 MHz. Therefore, theoretically, the frame rate of the main block is 368 fps (113 MHz/0.3072 M). The SDRAM controller's critical path clock period is 6.4 ns, which equals 156 MHz. However, because the SDRAM controller consists of 4 read/write ports, only 39 MHz can be achieved. Therefore, the frame rate is 127 fps (39 MHz/0.3072 M).

The proposed method was tested against this test set, and achieved the result shown in **Figure 13**.

4.2. Difference between BS and RTBS Equations

There is one well-known issue of background subtraction approach. If there is an object in the image at the beginning when the system starts to establish its background, the object will stay in the background for a period. This issue cloud also be a problem for the proposed RTBS method because the RTBS is an approximation of background subtraction methods. An experiment for RTBS concerning remain image is conducted, and analyzed with $N = 128$. During the FPGA experiment the frame number is shown by a seven-section monitor, which means a one is added to the seven-section monitor when the system completes the processing of one image. The experiment applies a man's hand as the background's foreign body, and maintains it for 20, 40, 60, 80 and 100 frames of 128, after which the hand object is removed. It is then noted how many frames it takes to clear up the object.

Results of the experiment are shown in **Figure 14**. The RTBS needs two frames to remove the hand object if the object appears in the beginning and lasts for 20 frames. With more lasting frames, the time to remove increases linearly. However, after 80 frames, it only needs around 10 image frames to clear up the object.

4.3. FPGA Hardware Resource Usage

FPGA resources include logic circuit and memory. Two frame buffers of about 998 KB of SDRAM memory are required external resources. **Table 3** shows the usage of FPGA resources for each function in the system. The analysis of resource utilization and memory requirement was calculated by using Altera QUARTUS II analyzer tool. QUARTUS II tool allows the user to launch Modelsim simulator from within the software using NativeLink. It facilitates the process of simulation by providing an easy to use mechanism and precompiled libraries for EDA (Electronic Design Automation) RTL (Register Transfer Level) and Gate-level Timing simulation.

(a) (b) (c)

Figure 13. Detection results of the RTBS. (a) Background frames, (b) Test frames, (c) Subtraction results.

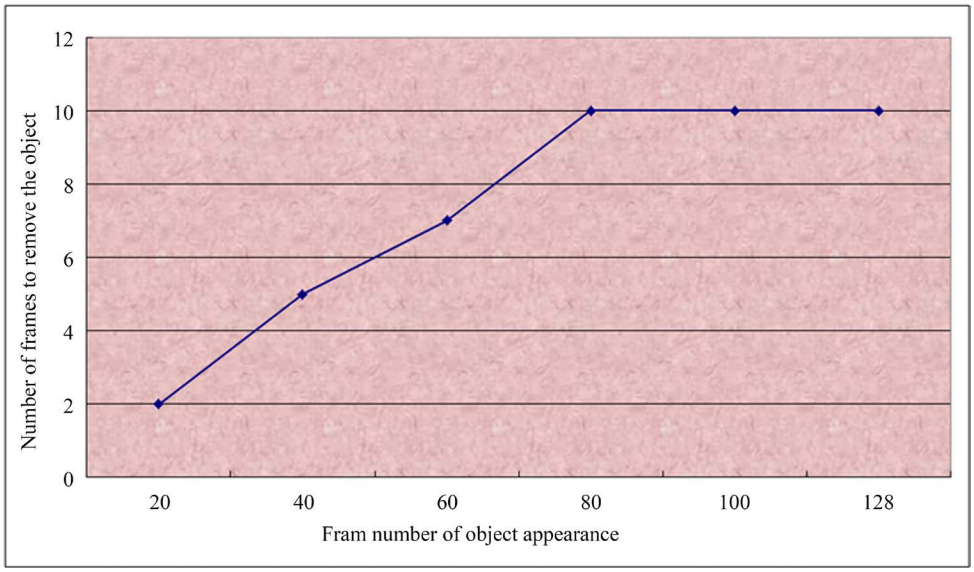

Figure 14. The linear relating between the lasting time of an object and the removal time of the object.

Table 3. Resource utilization in FPGA.

Module	Memory	Logic Circuit	
		Logic Element	Equivalent Gate Count
RTBS Module	0	154	1848
Morphology Module	7656	104	1248
RAW2RGB Module	12,800	88	1056
Mirror Module	19,200	21	252
RGB2Y Module	0	80	960
SDRAM Controller Module	52,348	828	9936
I/O and Other Modules	900	343	4116
Total Used Resource	92,904	1618	19,416

Generally speaking the LE (Logic Element) of ALTERA requires 8 ~ 21 logic gates. Typically it will be 12 logic gates [20]; internal memory usually requires 4 logic gates combined as 1 bit [20]. This study uses a typical value to estimate the design for the hardware's standard logic gates. **Table 4** shows the resource analysis index created for the DE2 Development board:

From the above index, it is determined that FPGA uses N4K internal memory, which is about equal to using 371,616 logic gates plus the BS equation, and other logic circuits use almost 19,416 logic gates. Therefore, the whole FPGA uses around 371,616 + 19,416 = 391,032 logic gates.

Finally, **Table 5** shows the realization of the proposed hardware performance compared with that achieved in past papers on Background Subtraction algorithms in FPGA.

5. Conclusion

This paper proposes a background subtraction and morphology algorithm accelerated by reconfigurable hardware which can help embedded systems achieve real-time security monitoring. The design partitions the functions into background modeling, subtraction and morphology. The high-cost function, background modeling, is reformulated by eliminating division operations, which both reduces resource utilization and improves performance. Data flow analysis further details the calculation of the design. In simulation, a high frame rate of 384 fps for the background subtraction with morphology and modeling module can be achieved at 25 Mhz for 640 × 480 resolution videos. Real-time performance of 51 fps for the whole system, including off-chip memory

Table 4. Using resource percentage in DE22C35FPGA.

Module	Memory		Logic Circuit		18×18 Multiplier	PLLs	I/O pin	8 MB SDRAM (bits)
	Block	Memory bit	Logic Element	Equivalent Gate Count				
Total DE2 Resource	105	483,840	33,216	398,592	35	4	475	67,108,864
Used Resource	36	92,904	1618	19,416	5	1	425	7,987,200
Used Percentage	34.29%	19.20%	4.87%	4.87%	14.29%	25.00%	89.47%	11.90%

Table 5. FPGA effectiveness and resource comparison table for past papers.

Ref	HW Algorithm	Device	Logic Circuit				Clock (MHz)	Performance			Power (w)
			LUTs	FF	B-RAM	DSP		Image Size	fps	Nfps*	
[12]	RTBS Module	Virtex II XC2v6000	No data	No data	No data	No data	25	720 × 576 monochromatic	60	27	No Data
[13]	Morphology Module	XC4010E	No data	No data	No data	No data	25	640 × 480	5	5	No Data
[8]	RAW2RGB Module	XCV800	No data	No data	No data	No data	25	640 × 480	15	15	0.121
[17]	Mirror Module	Virtex 5	1572	No data	No data	No data	47	1920 × 1080	22	79	0.027
[18]	RGB2Y Module	Virtex 2	No data	No data	No data	No data	40	1024 × 1024	38	81	No Data
[19]	SDRAM Controller Module	Virtex 6	788	363	0	3	189	1920 × 1080	91	81	0.0012
Our	I/O and Other Modules	Cyclone II	1618	0	36	5	25	640 × 480	56	56	0.0026
Our#	Total Used Resource	Cyclone II	1618	0	36	5	113	720 × 576	368	110	0.0026

Nfps*: Fps Normalized by 150 MHz clock and 1920 × 1080 Resolution; Our#: Our simulation results for 720 × 576 videos.

access, demonstrates the efficiency of the design. The implementation on low-end FPGA with low frequency indicates low power consumption. The final verification results show resource utilization of no more than 400 K gate counts, two frame buffers, and 1 MB SDRAM memory size. Further study of complex background subtraction algorithms such as Gaussian mixture model and LBP background subtraction is promising.

References

[1] Piccardi, M. (2004) Background Subtraction Techniques: A Review. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, The Hague, Netherlands, 10-13 October 2004, 3099-3104. http://dx.doi.org/10.1109/icsmc.2004.1400815

[2] Cucchiara, R., Grana, C., Piccardi, M. and Prati, A. (2003) Detecting Moving Objects, Ghosts, and Shadows in Video Streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**, 1337-1342. http://dx.doi.org/10.1109/TPAMI.2003.1233909

[3] Yang, Y.H. and Levine, M.D. (1992) The Background Primal Sketch: An Approach for Tracking Moving Objects. *Machine Vision and Applications*, **5**, 17-34.

[4] Masoud, O. and Papanikolopoulos, N.P. (2001) A Novel Method for Tracking and Counting Pedestrians in Real-time Using a Single Camera. *IEEE Transactions Vehicular Technology*, **50**, 1267-1278.

[5] Shoushtarian, B. and Bez, H.E. (2005) A Practical Adaptive Approach for Dynamic Backgrounds Subtraction Using an Invariant Colour Model and Object Tracking. *Pattern Recognition Letters*, **26**, 5-26.

[6] Wren, C.R., Azarbayejani, A., Darrell, T. and Pentland, A.P. (1997) Pfinder: Real-Time Tracking of the Human Body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**, 780-785. http://dx.doi.org/10.1109/34.598236

[7] Stauffer, C. and Grimson, W.E.L. (2000) Learning Patterns of Activity Using Real-Time Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**, 747-757.

[8] Elgammal, A., Duraiswami, R., Harwood, D. and Davis, L.S. (2002) Background and Foreground Modeling Using Nonparametric Kernel Density Estimation for Visual Surveillance. *Proceedings of the IEEE*, **90**, 1151-1163. http://dx.doi.org/10.1109/JPROC.2002.801448

[9] Chen, T.P., Horst, H., Alexander, B., Roman, B., Konstantin, R. and Alexander, K. (2005) Computer Vision Workload Analysis: Case Study of Video Surveillance Systems. *Intel Technology Journal*, **9**, 109-118.

[10] Ooi, M.P. (2006) Hardware Implementation for Face Detection on Xilinx Virtex-II FPGA Using the Reversible Component Transformation Colour Space. *Proceedings of the IEEE International Workshop on Electronic Design, Test and Applications*, Kuala Lumpur, 17-19 January 2006, 41-46.

[11] Bramberger, M., Brunner, J. and Rinner, B. (2004) Real Time Video Analysis on an Embedded Smart Camera for Traffic Surveillance. *Proceedings of the IEEE* 10*th Computer Society Conference on Real-Time and Embedded Technology and Applications Symposium*, Toronto, Canada, 25-28 May 2004, 174-181.

[12] Appiah, P., Hunter, K., Ormston, A. and Ormston, S. (2005) An FPGA-Based Infant Monitoring System. *Proceedings of the IEEE International Conference* on *Field-Programmable Technology*, Singapore, 11-14 December 2005, 315-316.

[13] Cucchiara, R., Onfiani, P., Prati, A. and Scarabottolo, N. (1999) Segmentation of Moving Objects at Frame Rate: A Dedicated Hardware Solution. *Proceedings of the IEEE* 7*th International Conference, Image Processing and Its Applications*, Manchester, UK, 12-15 July 1999, 138-142. http://dx.doi.org/10.1049/cp:19990297

[14] Stauffer, C. and Grimson, W.E.L. (1999) Adaptive Background Mixture Models for Real-time Tracking. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 246-252.

[15] Kaew TraKul Pong, P. and Bowden, R. (2001) An Improved Background Mixture Model for Real-time Tracking with Shadow Detection. *Proceedings of the* 2*nd European Workshop on Advanced Video Based Surveillance Systems*, London, UK, 7-10 September 2001, 135-144.

[16] Theocharides, T., Vijaykrishnan, N. and Irwin, M.J. (2006) A Parallel Architecture for Hardware Face Detection. *Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, Karlsruhe, 2-3 March 2006, 452-453. http://dx.doi.org/10.1109/ISVLSI.2006.10

[17] Genovese, M., Napoli, E. and Petra, N. (2010) OpenCV Compatible Real Time Processor for Background Foreground Identification. *Proceedings of the International Conference on Microelectronics*, Niš, Serbia, 16-19 May 2010, 467-470.

[18] Jang, H. Ardö, H. and Öwall, V. (2005) Hardware Accelerator Design for Video Segmentation with Multi-Modal Background Modelling. *Proc. IEEE Int. Symp. Circuits Syst.*, Japan, 23-26 May 2005, 1142-1145.

[19] Genovese, M. and Napoli, E. (2014) ASIC and FPGA Implementation of the Gaussian Mixture Model Algorithm for

Real-Time Segmentation of High Definition Video. *IEEE Transaction on Very Large Scale Integration (VLSI) System*, **22**, 537-547.

[20] Altera Corporation (1999) Gate Counting Methodology for APEX 20K Devices.